Attorney Docket No. IBM/182
Confirmation No. 4082

PATENT

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte William Joseph Armstrong, Chris Francois and Naresh Nayar

---

Appeal No. _____
Application No. 09/939,232

AMENDED APPEAL BRIEF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	William Joseph Armstrong et al.	Art Unit:	2123
Application No.:	09/939,232	Examiner:	Jason Scott Proctor
Filed:	August 23, 2001		
For:	SYSTEM FOR YIELDING TO A PROCESSOR		

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDED APPEAL BRIEF

I. REAL PARTY IN INTEREST

This application is assigned to International Business Machines Corporation, of Armonk, New York.

II. RELATED APPEALS AND INTERFERENCES

An appeal is currently pending in an application related to the instant application, U.S. Patent Application No. 09/939,235, also filed on August 24, 2001 by William Joseph Armstrong et al., entitled "YIELD ON MULTITHREADED PROCESSORS".

III. STATUS OF CLAIMS

Claims 1-19 and 21 are pending in the Application, stand rejected, and are now on appeal. Independent claims 1, 11 and 19 have each been amended twice, dependent claims 6, 9, 15 and 17-18 have each been amended once, claim 21 was added, and claim 20 has been canceled.

IV. STATUS OF AMENDMENTS

No amendments have been filed subsequent to the Final Rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Applicants' invention is generally directed to managing computer resources by more efficiently and effectively yielding virtual processors. The invention is more particularly directed to enabling a virtual processor to yield a CPU to another, "target" virtual processor to precipitate an occurrence upon which its own execution may be predicated (Application, page 5, lines 6-13).

A virtual processor is a programmatically simulated processor that may theoretically access many of the physical resources of the underlying physical machine. Exemplary resources may include memory assets and hardware registers, in addition to the CPU's. Virtual processors may additionally share a priority scheme or schedule that partially dictates allocation of processing cycles as between different virtual processors. An underlying program called a "hypervisor", or partition manager, may use this scheme to assign and dispatch CPU's to each virtual processor. For instance, the hypervisor may intercept requests for resources from operating systems to globally share and allocate them (Application, page 2, lines 6-1).

A yield command is a convention whereby virtual processors share access to CPU's. For instance, an idle virtual processor, e.g., with no work to do, may relinquish its CPU availability in response to receiving a yield command from another virtual processor with work to do. For example, a virtual processor may be in an idle loop or may have no work to do. The yield command causes the virtual processor to surrender its CPU availability to another virtual processor, and subsequently enter an idle state. Thus, the CPU is used more efficiently by virtue of its allocation to a virtual processor that can utilize it (Application, page 4, lines 1-10).

While the yielding virtual processors represents an improvement in managing resources, it nonetheless tolerates inefficiency. Namely, yielding virtual processors must often remain inactive for extended periods while waiting for the hypervisor to sequentially allocate CPU's to all other virtual processors. Even when allocated a CPU, a virtual processor's work may be predicated upon the work of another virtual processor that has yet to be allocated a CPU, resulting in the first virtual processor being unable to utilize the allocated CPU. Compounded over multiple processing layers and iterations, such inactivity translates into slow processing speeds and inefficiency (Application, page 4, lines 11-17).

Embodiments consistent with the principles of the present invention and addressing such problems include an apparatus, method, and program product configured to determine and designate a target virtual processor. An exemplary target processor may need a CPU held by the yielding virtual processor. The targeted processor may further embody a condition required prior to execution of the yielding processor. The yield feature of the embodiment may enable a hypervisor to allocate a CPU or memory resource surrendered by the yielding processor to the designated virtual processor. In enabling the target processor, the method of the embodiment may obviate the condition preventing execution of the yielding processor (Application, page 6, lines 12-22 through page 7, lines 1-5).

More particularly, an embodiment may initiate a request from a yielding virtual processor. The request may prompt a yield of a CPU controlled by the yielding virtual processor. The request may designate a target virtual processor from among the plurality of virtual processors. For instance, the signal may contain pointer and/or status information regarding the target processor (Application, page 5, lines 1-5).

The target virtual processor may require access to the CPU that the yielding virtual processor controls and requests to yield. The embodiment may then logically reassign, or "switch-in", control of the CPU from the yielding virtual processor to the target virtual processor. Prior to dispatching the CPU to the target virtual processor, the hypervisor may verify that the yielding virtual processor is still inactive, and that the target virtual processor is further not in a state of yield, itself. The hypervisor may further save the state of the yielding hypervisor prior to passing control to a dispatcher (Application, page 5, lines 6-13).

Specific support for the claimed subject matter for the independent claims as a whole has been provided above. Additional support for the claimed subject matter of the independent claims may also generally be found, for example, in Figs. 3 and 4, and at page 12, line 19-15 to page 18, line 22 of the Application as filed. However, a direct mapping of the aforementioned discussion to the individual independent claims, as requested by the Examiner in the Notification of Non-Compliant Appeal Brief dated August 8, 2006, is presented below:

Independent Claim 1

1. A method for yielding a virtual processor within a logically partitioned data processing system (page 4, lines 20 and 21, and Fig. 2, blocks 40, 42 and 44 of the Application), wherein the system supports a plurality of partitions (page 8, lines 11 and 12, and Fig. 2, blocks 40, 42 and 44 of the Application), a first of which includes a plurality of virtual processors used to schedule threads (page 4, line 22, and page 2, lines 9-11 of the Application) and that share at least one CPU (page 1, line 22 of the Application), and wherein the system further includes a hypervisor configured to assign and dispatch the CPU to the plurality of virtual processors (page 5, lines 9 and 10, and Fig. 2, block 46 of the Application), the method comprising:

requesting with a yielding virtual processor a yield of the CPU upon which the virtual processor is executing (page 5, lines 1 and 2, and Fig. 3, block 156 of the Application), including designating a target virtual processor from among the plurality of virtual processors (page 5, lines 3 and 4, and Fig. 3, block 156 of the Application); and

switching-in the target virtual processor for execution by the CPU in response to the requested yield (page 5, lines 8 and 9, and Fig. 3, block 165 of the Application).

Independent Claim 11

11. An apparatus comprising:

a logically partitioned computer including a plurality of logical partitions (page 8, lines 11 and 12, and Fig. 2, blocks 40, 42 and 44 of the Application), a first of which including a plurality of virtual processors used to schedule threads and that share at least one CPU (page 1, line 22; page 4, line 22; and page 2, lines 9-11 of the Application); and

a program resident in a hypervisor of the computer, wherein the program is configured to assign and dispatch the CPU to the plurality of virtual processors (page 5, lines 9 and 10, and Fig. 2, block 46 of the Application), the program configured to initiate a request for a yield of the CPU controlled by a yielding virtual processor (page 5, lines 1 and 2, and Fig. 3, block 156 of the Application), wherein the request designates a target virtual processor from among the plurality of virtual processors (page 5, lines 1 and 2, and Fig. 3, block 156 of the Application); and further

configured to logically reassign control of the CPU from the yielding virtual processor to the target virtual processor (page 5, lines 8 and 9, and Fig. 3, block 165 of the Application).

Independent Claim 19

19. A program product (page 11, lines 3 and 4 of the Application), comprising:

a program resident in a hypervisor configured to assign and dispatch a CPU to a plurality of virtual processors used to schedule threads (page 1, line 22; page 2, lines 9-11; page 4, line 22; page 5, lines 9 and 10; and Fig. 2, block 46 of the Application), wherein the program is further configured to initiate a request for a yield of the CPU controlled by a yielding virtual processor among a plurality of virtual processors in a logically partitioned data processing system (page 5, lines 1 and 2, and Fig. 3, block 156 of the Application), wherein the request designates a target virtual processor from among the plurality of virtual processors (page 5, lines 1 and 2, and Fig. 3, block 156 of the Application); and further configured to logically reassign control of the CPU from the yielding virtual processor to the target virtual processor (page 5, lines 8 and 9, and Fig. 3, block 165 of the Application); and

a tangible signal bearing medium bearing the first program (page 11, lines 6 through 9 of the Application).

Other features are recited in the dependent claims, and will be discussed in greater detail in the arguments section below. In addition, it should be noted that, as none of the claims recite any means plus function or step plus function elements, no identification of such elements is required pursuant to 37 CFR §41.37(c)(1)(v), nor is any summary of the claimed subject matter of any separately argued dependent claim required.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 1-19 and 21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,404,563 to Green et al. (*Green*) in view of U.S. Patent No. 5,872,963 to Bitar et al. (*Bitar*) and further in view of *Operating System Concepts, Fifth Edition* by Abraham Silberschatz and Peter Baer Galvin (*Silberschatz*).

VII. ARGUMENT

In summary, the pending claims are directed to yielding, or switching-in virtual processors. The Examiner admits that his primary reference discloses only switching between threads (not virtual processors), but the Examiner erroneously asserts that threads can be the equivalent of virtual processors. Equating a virtual processor to a thread is contrary to the known meaning of the terms, the express language of the primary reference, itself, and even the previous position of the Examiner. Moreover, such a misrepresentation is non sequitur in terms of the language of the presently pending claims. Because a virtual processor is patentably distinct from a thread, the prior art fails to suggest or motivate anything more than thread switching. Applicants consequently submit that the Examiner's rejections of claims 1-19 and 21 are not supported on the record, and should be reversed.

Applicants will hereinafter address the Examiner's rejections in the order presented in the Final Office Action. Within the discussion of each rejection, the various claims that are the subject of the Examiner's rejections will further be addressed in order, starting with the independent claims, and then addressing various dependent claims reciting additional subject matter that is distinguishable from the prior art of record. In some cases, specific discussions of particular claims are not made in the interests of streamlining the appeal. The omission of a discussion with respect to any particular claim, however, should not be interpreted as an acquiescence as to the merits of the Examiner's rejection of the claim, particularly with respect to claims reciting features that are addressed in connection with the rejections applied to other claims pending in the appeal.

A. Claims 1-19 and 21 are non-obvious over *Green* in view of *Bitar* and in view of *Silberschatz*.

The Examiner asserts that all of the claims, which are directed to yielding, designating and switching-in virtual processors, are obvious over a combination of *Green*, *Bitar* and *Silberschatz*. Yet none of the cited art discloses or suggests those claimed features regarding virtual processors, as would be required to justify such an assertion.[1] Consequently and as discussed in detail below, the rejections should be reversed.

Independent Claims 1, 11 and 19

The Examiner argues that the combination of *Green*, *Bitar* and *Silberschatz* render claims 1, 11 and 19 obvious. Applicants respectfully submit, however, that the cited prior art fails to motivate, suggest or anywhere disclose the features recited in these claims, e.g., designating and switching-in a virtual processor, and therefore the Examiner's rejection under 35 U.S.C. §103(a) should be reversed. Applicants will discuss method claim 1 prior to discussing claims 11 and 19 (directed respectively to an apparatus and a program product).

Claim 1 generally recites a method for yielding a virtual processor within a logically partitioned data processing system, wherein the system supports a plurality of partitions, a first of which includes a plurality of virtual processors that share at least one CPU and that are used to schedule threads. The system includes a hypervisor configured to assign and dispatch the CPU to the plurality of virtual processors. The method further comprises requesting with a yielding virtual processor a yield of the CPU upon which the virtual processor is executing, including designating a target virtual processor from among the plurality of virtual processors. The method

---

[1] A *prima facie* showing of obviousness requires that the Examiner establish that the differences between a claimed invention and the prior art "are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art." 35 U.S.C. §103(a). Such a showing requires that all claimed features be disclosed or suggested by the prior art, along with objective evidence of the suggestion, teaching or motivation to combine or modify prior art references, as "[c]ombining prior art references without evidence of such a suggestion, teaching or motivation simply takes the inventor's disclosure as a blueprint for piecing together the prior art to defeat patentability -- the essence of hindsight." In re Dembiczak, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999).

also includes switching-in the target virtual processor for execution by the CPU in response to the requested yield.

The Examiner admits that *Bitar* discloses only switching between threads (not virtual processors). He further concedes that *Green* merely provides background knowledge on hypervisors,[2] and that *Silberschatz* is cited solely to show "an example of the inherency of using a *virtual processor* to schedule *threads* " (emphasis added).[3] As such, no motivation or suggestion regarding virtual processors can properly be said to be found in the cited prior art.

To support his position, however, the Examiner erroneously asserts that threads are the equivalent of virtual processors.[4] That is, the basis of Examiner's argument is that the thread switching processes of *Bitar* are the same as the claimed virtual processor steps. This position of equating a virtual processor to a thread is contrary to the known meaning of the terms, the express language of the primary reference, itself, and even the previous position of the Examiner.

The assertion of the Examiner is especially discouraging because *Bitar*, itself, dedicates more than a paragraph to distinguishing between threads and the claimed virtual processors: "Conceptually, it is useful to distinguish the user-level thread from the virtual processor" (col. 1, lines 27-33). Basically, *Bitar* explains this distinction as being between (user) threads ("the instantiation of each activity during execution" - col. 1, lines 16-19) and virtual processors ("the basic unit of scheduling" - col. 1, line 56).

Another instance in *Bitar* that plainly contradicts the Examiner's assertion that threads and virtual processor are equivalent includes Fig. 2b, which distinctly shows threads 2 mapped to kernel space comprising virtual processors 5 (col. 5, lines 64-65). Fig. 8 likewise shows threads 24.M switched without using kernel space 18 (and associated virtual processors 45).

In the context of *Bitar*, this distinction between user threads and virtual processors (or kernel threads in a Mach system, col. 1, lines 54-59 and col. 2, line 63) is important because *Bitar* is focused on switching threads in a way that does not burden virtual processors. The objective of *Bitar* is to achieve switching between user threads without involving the scheduling

---

[2] Final Office Action, dated 02/15/06, at pg. 4.

[3] *Id.* at pg. 9.

[4] *Id.* at pg. 4.

of virtual processors (col. 5, lines 31-33 and lines 55-58). *Bitar* teaches away from using a virtual processor, or kernel, and associated scheduling during thread switching for efficiency reasons (col. 5, lines 14-18, 31-38, 55-58 and col. 12, line 24).

When this text of *Bitar* was pointed out to the Examiner, his response in the Final Office Action hinged on a bizarre misinterpretation of the above quoted *Bitar* text, asserting that the distinction drawn in *Bitar* is "merely a conceptual" distinction and not a "plain" one.[5] The Examiner proceeded in the Action to speculate why *Bitar* might may have wished for the distinction to be conceptual, drawing from a quotation that conspicuously omits contextual reference to a virtual processor's function of mapping user-level threads.[6]

Notwithstanding that "conceptual" is not an antonym for "plain", this stance of the Examiner is a metaphor for his entire argument: ignoring explicit language to make assumptions contrary to the plain meaning of the text.

Perhaps the Examiner's confusion has arisen from *Bitar* text that describes a *kernel* thread in a Mach operating system as being the equivalent of a virtual processor in other operating systems (column 1, lines 54-column 2, line 5). However, *Bitar* characterizes both the kernel thread and the virtual processor as being "the basic unit of scheduling". What these units are being used to schedule, or map, are *user* threads. Of note, claim 1 explicitly recites the function of the virtual processor as a entity used for scheduling. Applicants consequently submit that the functional distinction between a virtual processor and a thread is patentably clear.

As part of his basis for asserting that a virtual processor can be a user thread, the Examiner asserts that an "execution entity" can be a virtual processor in terms of the disclosure of *Bitar*.[7] As discussed above, however, *Bitar* describes only threads as execution entities, in contrast with scheduling entities, such as virtual processors (column 1, lines 16-19 and 56). As such, Examiner's attempt to expand *Bitar*'s definition of execution entities to include virtual processors is improper and contrary to the plain text of *Bitar*.

---

[5] *Id.* at page 6.

[6] *Bitar* at col. 1, lines 27-29.

[7] Final Office Action, dated 02/15/06 at pg. 4.

In any case, as presently claimed, virtual processors are different than threads. The express claim language recites the function of a virtual processor for use in scheduling thread execution. Conventional operating systems dispatch threads (units of execution) to CPU's. In a logically partitioned system, the operating system thinks that virtual processors are CPU's for purposes of dispatching threads. That is, the operating system dispatches threads to virtual processors. The hypervisor schedules CPU resources to the virtual processors to execute the threads dispatched to the virtual processors. The virtual processor is used (by the hypervisor) to schedule thread execution. As such, the Examiner's assertion that a virtual processor is equivalent to a thread is contrary to the known meaning of the words, the express claim language, and the terms as defined in the cited prior art.

Green provides only background knowledge and does not disclose the claimed scheduling technique.[8] Moreover, the Examiner also acknowledges that there is no hypervisor in *Bitar*[9] (because *Bitar* is trying to avoid scheduling associated with hypervisors).

Since no cited reference discloses or suggests switching-in virtual processors, the rejections should be reversed. Reversal of the Examiner's rejection of claim 1 is therefore respectfully requested.

Next, with respect to independent claims 11 and 19, which respectively recite an apparatus and a program product, each of these claims likewise recite method for yielding a virtual processor within a logically partitioned data processing system, wherein the system supports a plurality of partitions, a first of which includes a plurality of virtual processors that share at least one CPU and that are used to schedule threads. The claims include processes for requesting with a yielding virtual processor a yield of the CPU upon which the virtual processor is executing, including designating a target virtual processor from among the plurality of virtual processors. Claimed processes also include switching-in the target virtual processor for execution by the CPU in response to the requested yield. Accordingly, Applicants respectfully submit that claims 11 and 19 are novel and non-obvious over the cited prior art of record for the

---

[8] *Id.*

[9] Non-Final Office Action of 09/06/05, at pg. 2.

same reasons as claim 1. Reversal of the Examiner's rejections, and allowance of claims 11 and 19, are therefore respectfully requested.

Dependent Claims 2 and 12

Dependent claims 2 and 12 respectively depend from claims 1 and 11, and each recites that the target virtual processor requires access to the CPU, and the yielding virtual processor controls the CPU. In rejecting these claims, the Examiner asserts that a virtual processor accomplished the claimed features. As discussed herein, however, the cited prior art does not suggest or motivate virtual processors, let alone that a target virtual processor requiring CPU access. Reversal of the Examiner's rejections, and allowance of claims 2 and 12, are therefore respectfully requested.

Dependent Claims 3 and 13

Dependent claims 3 and 18 respectively depend from claims 1 and 11, and each recites generating a yield command from the virtual processor, wherein the yield command includes pointer and status information regarding the target virtual processor. Again, the cited prior art fails to motivate or suggest the claimed virtual processors or a yield command, let alone the recited pointer and status features. Reversal of the Examiner's rejections, and allowance of the claims, are therefore respectfully requested.

Dependent Claim 4

Dependent claims 4 depends from claim 1 and regards assigning status information to the target virtual processor. The cited prior art fails to suggest or motivate virtual processor yield processes, let alone the claims assignment features. As such, Applicants respectfully submit that the Examiner has failed to establish anticipation of claim 4. Reversal of the Examiner's rejection, and allowance of the claim, are therefore respectfully requested.

Dependent Claims 5 and 14

Dependent claims 5 and 14 respectively depend from claims 1 and 11, and each regards assigning a target count to the target virtual processor. The cited prior art fails to suggest or motivate virtual processor yield processes, let alone the claims assignment features. As such,

Applicants respectfully submit that the Examiner has failed to establish anticipation of claims 5 and 14. Reversal of the Examiner's rejections, and allowance of the claims, are therefore respectfully requested.

Dependent Claims 6 and 15

Dependent claims 6 and 15 respectively depend from claims 5 and 14, and each recites comparing the target count to a presented count conveyed in the requested yield. The cited prior art neither suggests nor motivates virtual processors, let alone target count comparisons are therefore respectfully requested.

Dependent Claims 7 and 16

Dependent claims 7 and 16 respectively depend from claims 1 and 11, and each recites aborting the yield in response to a yield-to-active command. The cited prior art does not disclose, motivate or suggestion yield commands, let alone virtual processors. As such, reversal of the Examiner's rejections, and allowance of the claims, are therefore respectfully requested.

Dependent Claims 8 and 17

Dependent claims 8 and 17 respectively depend from claims 1 and 11, and each recites designating the yielding virtual processor as waiting for the target virtual processor. The cited prior art neither suggests nor motivates virtual processors. Applicants consequently request reversal of the Examiner's rejections, and allowance of the claims, are therefore respectfully requested.

Dependent Claims 9 and 18

Dependent claims 9 and 18 respectively depend from claims 1 and 11, and each recites designating the target virtual processor as having a yielding and waiting virtual processor. The cited prior art neither suggests nor motivates virtual processors. Applicants consequently request reversal of the Examiner's rejections, and allowance of the claims, are therefore respectfully requested.

Dependent Claim 10

Dependent claim 10 depends from claim 1, and recites storing the state of the yielding virtual processor. The cited prior art neither suggests nor motivates virtual processors. Applicants consequently request reversal of the Examiner's rejection, and allowance of the claim, are therefore respectfully requested.

Dependent Claim 21

Dependent claim 21 depends from claim 11, and recites a schedule used to determine allocation as between a plurality of virtual processors. The cited prior art neither suggests nor motivates virtual processors. *Bitar* teaches actually away from virtual processor scheduling, as discussed above. Applicants consequently request reversal of the Examiner's rejection, and allowance of the claim, are therefore respectfully requested.

CONCLUSION

In conclusion, Applicants respectfully request that the Board reverse the Examiner's rejections of claims 1-19 and 21, and that the Application be passed to issue. If there are any questions regarding the foregoing, please contact the undersigned at 513/241-2324. Moreover, if any other charges or credits are necessary to complete this communication, please apply them to Deposit Account 23-3000.

Respectfully submitted,

WOOD, HERRON & EVANS, L.L.P.

Date: September 8, 2006

By: /Douglas A. Scholer/
Douglas A. Scholer
Reg. No. 52,197

2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
(513) 241-2324 - Telephone
(513) 241-6234 - Facsimile

VIII. CLAIMS APPENDIX: CLAIMS ON APPEAL (S/N 09/939,232)

1. A method for yielding a virtual processor within a logically partitioned data processing system, wherein the system supports a plurality of partitions, a first of which includes a plurality of virtual processors used to schedule threads and that share at least one CPU, and wherein the system further includes a hypervisor configured to assign and dispatch the CPU to the plurality of virtual processors, the method comprising:

requesting with a yielding virtual processor a yield of the CPU upon which the virtual processor is executing, including designating a target virtual processor from among the plurality of virtual processors; and

switching-in the target virtual processor for execution by the CPU in response to the requested yield.

2. The method according to claim 1, wherein the target virtual processor requires access to the CPU, wherein the yielding virtual processor controls the CPU.

3. The method according to claim 1, further comprising generating a yield command from the virtual processor, wherein the yield command includes pointer and status information regarding the target virtual processor.

4. The method according to claim 1, further comprising assigning status information to the target virtual processor.

5. The method according to claim 1, further comprising assigning a target count to the target virtual processor.

6. The method according to claim 5, further comprising comparing the target count to a presented count conveyed in the requested yield.

7. The method according to claim 1, further comprising aborting the yield in response to a yield-to-active command.

8. The method according to claim 1, further comprising designating the yielding virtual processor as waiting for the target virtual processor.

9. The method according to claim 1, further comprising designating the target virtual processor as having a yielding processor waiting for the yielding virtual processor.

10. The method according to claim 1, further comprising storing the state of the yielding virtual processor.

11. An apparatus comprising:

a logically partitioned computer including a plurality of logical partitions, a first of which including a plurality of virtual processors used to schedule threads and that share at least one CPU; and

a program resident in a hypervisor of the computer, wherein the program is configured to assign and dispatch the CPU to the plurality of virtual processors, the program configured to initiate a request for a yield of the CPU controlled by a yielding virtual processor, wherein the request designates a target virtual processor from among the plurality of virtual processors; and further configured to logically reassign control of the CPU from the yielding virtual processor to the target virtual processor.

12. The apparatus according to claim 11, wherein the target virtual processor requires access to the CPU, wherein the yielding virtual processor controls the CPU.

13. The apparatus according to claim 11, wherein the program initiates generation of a yield command from the virtual processor, wherein the yield command includes pointer and status information regarding the target virtual processor.

14. The apparatus according to claim 11, wherein the program initiates an assignment of a target count to the target virtual processor.

15. The apparatus according to claim 14, wherein the program initiates a comparison of the target count to a presented count conveyed in the request for the yield.

16. The apparatus according to claim 11, wherein the program initiates abandonment of the yield in response to a yield-to-active command.

17. The apparatus according to claim 11, wherein the program initiates a designation of the yielding virtual processor as waiting for the target virtual processor.

18. The apparatus according to claim 11, wherein the program designates the target virtual processor as having a yielding processor waiting for the yielding virtual processor.

19. A program product, comprising:

a program resident in a hypervisor configured to assign and dispatch a CPU to a plurality of virtual processors used to schedule threads, wherein the program is further configured to initiate a request for a yield of the CPU controlled by a yielding virtual processor among a plurality of virtual processors in a logically partitioned data processing system, wherein the request designates a target virtual processor from among the plurality of virtual processors; and further configured to logically reassign control of the CPU from the yielding virtual processor to the target virtual processor; and

a tangible signal bearing medium bearing the first program.

21. The apparatus according to claim 11, wherein at least one of the virtual processors of the plurality of virtual processors includes a schedule used to determine allocation as between the plurality of virtual processors of processing cycles of the CPU.

IX. EVIDENCE APPENDIX

09/939,232

None.

X. RELATED PROCEEDINGS APPENDIX

09/939,232

None.